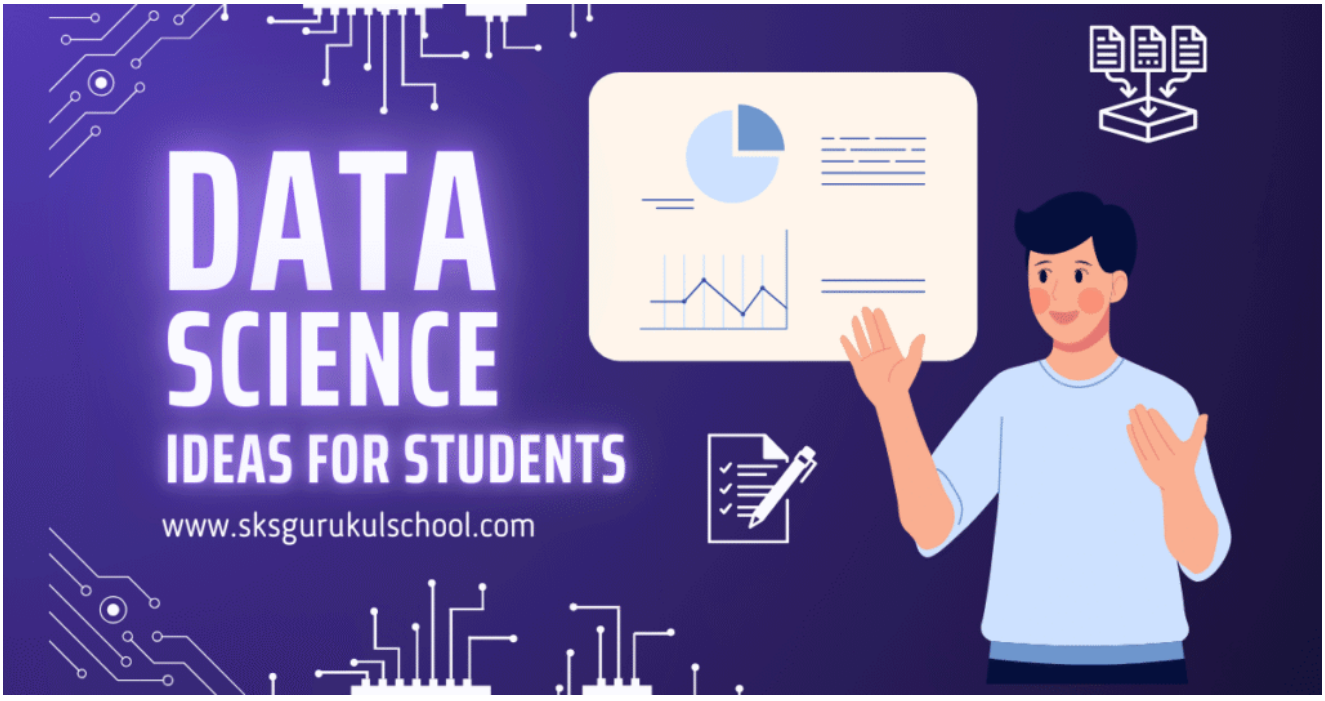
30 Data Science Project Ideas for Students 2026-27



Data science is one of the most practical and high-impact fields students can learn today. Working on projects is the fastest way to turn theory into real skills.

Projects help you practice data cleaning, visualization, modeling, evaluation, and communication — all essential parts of being a data scientist.

This article gives you **30 detailed, student-friendly data science project ideas**.

Each idea includes a clear objective, suggested tools and datasets, step-by-step implementation guidance, expected learning outcomes, difficulty level, and possible extensions.

You can pick projects based on your current skill level and the time you have. Read the intro, pick a few projects that excite you, and start building — that portfolio project will make your resume stand out.

How to use this list

1. Start with projects marked *Beginner* if you are new to Python and data analysis.
2. Choose *Intermediate* projects after you know pandas, matplotlib/seaborn, and a bit of scikit-learn.
3. Move to *Advanced* projects once you understand model selection, feature engineering, and basic deep learning.
4. For every project: collect data, explore it, clean it, analyze or model it, evaluate results, and write a clear README or blog post describing your process and findings.
5. Aim to put each finished project on GitHub and include a short demo or screenshots.

Must Read: 30 AI Project Ideas for Class 10

30 Data Science Project Ideas for Students 2026-27

1. Student Performance Analysis (Beginner)

Objective: Analyze factors affecting student exam performance and predict grades.
Tools: Python, pandas, matplotlib/seaborn, scikit-learn.
Dataset suggestions: UCI Student Performance dataset, school datasets.
Steps:

1. Load data and inspect columns (demographics, study time, absences, grades).
2. Clean missing values and encode categorical variables.
3. Perform exploratory data analysis: correlations, boxplots, and distributions.
4. Build baseline models: linear regression or decision tree for grade prediction.
5. Evaluate using RMSE/MAE and cross-validation.
6. Visualize influential features and write findings.
   Learning outcomes: EDA, basic regression, feature encoding.
   Difficulty: Beginner.
   Extensions: Create a web dashboard showing student risk indicators.

2. Movie Recommendation (Collaborative Filtering) (Intermediate)

Objective: Build a simple movie recommender using collaborative filtering.
Tools: Python, pandas, surprise or implicit library, Flask for demo.
Dataset: MovieLens (100k or 1M).
Steps:

1. Load user-item ratings and create user-item matrix.
2. Implement user-based or item-based collaborative filtering (k-NN).
3. Evaluate with RMSE or precision@K using train-test split.
4. Build a simple interface where a user inputs liked movies and receives recommendations.
5. Explain cold-start issues and potential improvements.
   Learning outcomes: Recommendation systems basics, similarity metrics, model evaluation.
   Difficulty: Intermediate.
   Extensions: Hybrid recommender using content features (genre, year).

3. Twitter Sentiment Analysis (Intermediate)

Objective: Classify tweet sentiment (positive, negative, neutral) about a topic.
Tools: Python, tweepy (or pre-collected data), pandas, nltk/spacy, scikit-learn, or transformers.
Dataset: Kaggle Twitter sentiment datasets or collect via API.
Steps:

1. Collect tweets (or use dataset) and clean (remove URLs, mentions).
2. Tokenize, remove stopwords, and apply lemmatization.
3. Transform using TF-IDF or word embeddings.
4. Train classification models: Logistic Regression, SVM, or a simple BERT fine-tune.
5. Evaluate with F1-score and confusion matrix.
6. Visualize common positive/negative words and emotion trends over time.
   Learning outcomes: Text preprocessing, NLP pipelines, classification metrics.
   Difficulty: Intermediate.
   Extensions: Sentiment time series monitoring or aspect-based sentiment analysis.

4. Sales Forecasting with Time Series (Advanced)

Objective: Forecast future sales for a product or store.
Tools: Python, pandas, Prophet or statsmodels, scikit-learn.
Dataset: Retail sales datasets (Kaggle: Rossmann store sales).
Steps:

1. Aggregate sales data by date and handle missing dates.
2. Visualize seasonality, trends, and calendar effects.
3. Try baseline models: moving average, ARIMA, then advanced: Prophet or LSTM.

4. Evaluate forecasting accuracy using MAPE or RMSE.
5. Incorporate external features (promotions, holidays).
6. Create forecast visualizations and a short report.

Learning outcomes: Time series decomposition, forecasting models, feature engineering for time.
Difficulty: Advanced.
Extensions: Multi-store forecasting or probabilistic forecasts.

5. House Price Prediction (Intermediate)

Objective: Predict house prices using structured features.
Tools: Python, pandas, scikit-learn, XGBoost/LightGBM.
Dataset: Kaggle House Prices – Advanced Regression Techniques.
Steps:

1. Load and understand the features (area, year built, neighborhood).
2. Handle missing values and encode categorical variables.
3. Do feature engineering (age, total rooms).
4. Train and compare models: linear regression, random forest, XGBoost.
5. Use cross-validation and evaluate with RMSE.
6. Explain important features using SHAP or feature importances.

Learning outcomes: Regression modeling, feature engineering, model interpretation.
Difficulty: Intermediate.
Extensions: Build a simple web form to estimate house price.

6. Image Classification (Beginner to Intermediate)

Objective: Classify images into categories (e.g., cats vs dogs).
Tools: Python, TensorFlow/Keras or PyTorch, OpenCV.
Dataset: Kaggle Cats & Dogs, CIFAR-10.
Steps:

1. Load images and preprocess (resize, normalize).
2. Split into train/validation/test sets.
3. Train a simple CNN from scratch or use transfer learning (MobileNet, ResNet).
4. Monitor training and avoid overfitting with augmentation and dropout.
5. Evaluate with accuracy and confusion matrix.
6. Save the model and create a demo script that classifies uploaded images.

Learning outcomes: Deep learning basics, CNNs, transfer learning.
Difficulty: Beginner–Intermediate.
Extensions: Multi-class classification or explainability techniques.

7. Credit Card Fraud Detection (Advanced)

Objective: Detect fraudulent transactions in imbalanced datasets.
Tools: Python, pandas, scikit-learn, imbalanced-learn, XGBoost.
Dataset: Credit Card Fraud Detection dataset (Kaggle).
Steps:

1. Understand class imbalance and data features.
2. Do EDA and visualize class distributions.
3. Try resampling techniques: SMOTE, undersampling, or class weights.
4. Train models: logistic regression, random forest, XGBoost.
5. Evaluate with precision-recall curves, F1-score, and AUC-PR.
6. Discuss real-world deployment considerations.
   Learning outcomes: Imbalanced learning, model evaluation beyond accuracy, risk assessment.
   Difficulty: Advanced.
   Extensions: Real-time detection pipeline simulation.

8. Customer Segmentation (K-Means Clustering) (Intermediate)

Objective: Segment customers for targeted marketing.
Tools: Python, pandas, scikit-learn, matplotlib.
Dataset: Online retail datasets (UCI Online Retail).
Steps:

1. Clean transaction data and compute RFM features (Recency, Frequency, Monetary).
2. Scale features and apply K-Means clustering.
3. Use elbow and silhouette methods to select k.
4. Profile each segment and interpret business meaning.
5. Visualize clusters and present marketing actions for each group.
   Learning outcomes: Unsupervised learning, feature scaling, business interpretation.
   Difficulty: Intermediate.
   Extensions: Use hierarchical clustering or DBSCAN.

9. Traffic Accident Analysis & Hotspot Detection (Intermediate)

Objective: Identify accident hotspots and factors contributing to accidents.
Tools: Python, geopandas, folium, scikit-learn.
Dataset: Local government open data on accidents or Kaggle datasets.
Steps:

1. Load location-based accident data and clean timestamps.
2. Map accidents and use density estimation to find hotspots.

3. Analyze features like weather, time of day, and vehicle types.
4. Build a model to predict accident severity or high-risk times.
5. Create map visualizations showing hotspots and recommendations.

Learning outcomes: Geospatial analysis, mapping, density estimation.
Difficulty: Intermediate.
Extensions: Real-time dashboard or predictions by area.

10. Chatbot with FAQ Retrieval (Beginner)

Objective: Build a retrieval-based chatbot for a specific domain (college FAQ).
Tools: Python, nltk/spacy, scikit-learn, Flask.
Dataset: Manually collect FAQs or use existing FAQ pages.
Steps:

1. Prepare question-answer pairs and clean text.
2. Use TF-IDF vectorization for query and FAQ embeddings.
3. For a user query, find the most similar FAQ using cosine similarity.
4. Return the best matching answer and handle low-confidence responses.
5. Build a simple web interface for the chatbot.

Learning outcomes: Text similarity, retrieval systems, simple web deployment.
Difficulty: Beginner.
Extensions: Add a fallback to open-ended generative responses.

11. Healthcare: Disease Prediction (Intermediate)

Objective: Predict presence or risk of a disease (e.g., diabetes).
Tools: Python, pandas, scikit-learn.
Dataset: Pima Indians Diabetes dataset (UCI), heart disease datasets.
Steps:

1. Explore features and understand clinical meaning.
2. Preprocess and scale features.
3. Train classification models and calibrate probabilities.
4. Evaluate with ROC-AUC and confusion matrix.
5. Discuss ethical considerations and potential biases.

Learning outcomes: Working with medical datasets, model evaluation, ethics.
Difficulty: Intermediate.
Extensions: Build a risk score calculator and a patient-facing report.

12. Hands-on Exploratory Data Analysis Portfolio (Beginner)

Objective: Create EDA-focused projects on multiple small datasets to demonstrate analysis skills.

Tools: Python, pandas, matplotlib/seaborn, Jupyter Notebook.
Dataset: Titanic, Iris, Wine Quality, and others on Kaggle.
Steps:

1. For each dataset, perform thorough EDA: missing values, outliers, trends.
2. Use visuals to tell a story and derive insights.
3. Summarize findings in a clear notebook with markdown explanations.
4. Organize notebooks in a portfolio with README.
   Learning outcomes: Strong EDA skills and communication.
   Difficulty: Beginner.
   Extensions: Add predictive models in some notebooks.

13. Topic Modeling on Articles (Intermediate)

Objective: Discover themes in a corpus using topic modeling.
Tools: Python, gensim, nltk/spacy, pyLDAvis.
Dataset: News articles, blog posts, or research abstracts.
Steps:

1. Collect and clean text corpus.
2. Preprocess (tokenize, lemmatize, remove stopwords).
3. Build LDA or NMF topic models and tune the number of topics.
4. Visualize topics with pyLDAvis and label them.
5. Use topics to cluster or summarize new articles.
   Learning outcomes: Unsupervised NLP, model selection, topic visualization.
   Difficulty: Intermediate.
   Extensions: Use dynamic topic modeling over time.

14. Predicting Bike Sharing Demand (Time Series/Regression) (Advanced)

Objective: Predict hourly/daily bike rental demand.
Tools: Python, pandas, scikit-learn, LightGBM, Prophet.
Dataset: Bike Sharing dataset (UCI / Capital Bikeshare).
Steps:

1. Aggregate data by hour and include weather and calendar features.
2. Visualize seasonal and daily patterns.
3. Train regression models with lag features and rolling statistics.
4. Evaluate forecasts using RMSE and MAPE.
5. Propose operational actions (rebalancing bikes).
   Learning outcomes: Time series feature engineering, applied forecasting.

Difficulty: Advanced.
Extensions: Real-time prediction API for dock suggestion.

15. Fake News Detection (Advanced)

Objective: Classify news articles as real or fake.
Tools: Python, scikit-learn, transformers (BERT), nltk.
Dataset: LIAR dataset, FakeNewsNet, or Kaggle Fake News datasets.
Steps:

1. Collect labeled articles and clean text.
2. Use TF-IDF and classical models as baseline.
3. Fine-tune a transformer-based classifier (BERT) for better accuracy.
4. Evaluate using precision/recall and analyze false positives.
5. Discuss societal impact and limitations.
   Learning outcomes: NLP classification, transfer learning, ethics.
   Difficulty: Advanced.
   Extensions: Provide explainability for predictions.

16. Energy Consumption Forecasting (Intermediate)

Objective: Forecast household or building energy usage.
Tools: Python, pandas, statsmodels, Prophet.
Dataset: UCI Individual household electric power consumption.
Steps:

1. Clean time series data and handle missing values.
2. Visualize daily and weekly cycles and perform decomposition.
3. Try ARIMA, Prophet, and tree-based regressors with lag features.
4. Evaluate and visualize forecasts.
5. Suggest energy saving strategies based on patterns.
   Learning outcomes: Time series preprocessing and modeling.
   Difficulty: Intermediate.
   Extensions: Build an alert system for abnormal consumption.

17. Image Segmentation for Simple Objects (Advanced)

Objective: Segment objects (e.g., road signs) from images.
Tools: Python, TensorFlow/Keras or PyTorch, OpenCV.
Dataset: Pascal VOC, COCO (or smaller custom dataset).
Steps:

1. Prepare dataset with masks and train-validation split.
2. Implement U-Net or use pre-trained segmentation models.

3. Train with appropriate loss (Dice/IoU).
4. Evaluate segmentation accuracy and visualize masks.
5. Optimize model size for faster inference.

Learning outcomes: Computer vision, segmentation techniques, performance metrics.
Difficulty: Advanced.
Extensions: Deploy to mobile or web for real-time segmentation.

18. Natural Language Question Answering (Intermediate to Advanced)

Objective: Build a system that answers domain-specific questions from documents.
Tools: Python, transformers (BERT-based QA models), Flask.
Dataset: SQuAD for practice; use company manuals or a collection of PDFs for domain-specific QA.
Steps:

1. Index documents and split into chunks.
2. Use a pre-trained QA model to extract answers from chunks.
3. Rank answers by confidence and present best match.
4. Build a simple UI for question input.
5. Evaluate using exact match and F1 on a test set.

Learning outcomes: Transformer-based QA, document chunking, retrieval+reader pipelines.
Difficulty: Intermediate–Advanced.
Extensions: Add retrieval using dense embeddings (FAISS).

19. Forecasting Stock Returns (Advanced, caution)

Objective: Build a model to forecast short-term stock returns (educational only).
Tools: Python, pandas, scikit-learn, LightGBM.
Dataset: Yahoo Finance historical data.
Steps:

1. Compute technical indicators as features (moving averages, RSI).
2. Avoid look-ahead bias by careful feature engineering.
3. Train classification/regression models to predict next-day direction.
4. Evaluate using backtesting and metrics that account for transaction costs.
5. Discuss risks and disclaimers — this is for learning, not financial advice.

Learning outcomes: Financial feature engineering, backtesting, model limitations.
Difficulty: Advanced.
Extensions: Paper-trade a strategy with strict risk management.

20. Voice Gender Recognition (Beginner to Intermediate)

Objective: Predict speaker gender from audio features.
Tools: Python, librosa for audio features, scikit-learn.
Dataset: Free voice datasets (VCTK or Kaggle voice gender).
Steps:

1. Extract features like MFCCs, chroma, and spectral contrast.
2. Aggregate features and normalize.
3. Train classification models (SVM, Random Forest).
4. Evaluate with accuracy and confusion matrix.
5. Build a small script to record audio and predict gender.
   Learning outcomes: Audio feature extraction, classification.
   Difficulty: Beginner–Intermediate.
   Extensions: Add speaker identification or emotion detection.

21. Automated Resume Screening (Intermediate)

Objective: Rank or classify resumes against a job description.
Tools: Python, spaCy, scikit-learn, transformers.
Dataset: Public resume/job posts or create synthetic data.
Steps:

1. Parse resumes into structured fields (skills, education).
2. Vectorize job descriptions and resumes with embeddings.
3. Compute similarity score and rank candidates.
4. Build rule-based filters for must-have skills.
5. Evaluate ranking quality with human-labeled pairs.
   Learning outcomes: Information extraction, semantic matching, ranking.
   Difficulty: Intermediate.
   Extensions: Add diversity-aware scoring and bias mitigation.

22. Sales Conversion Funnel Analysis (Beginner to Intermediate)

Objective: Analyze user funnel and suggest improvements to increase conversions.
Tools: Python, pandas, matplotlib, Google Analytics data (if available).
Dataset: Sample e-commerce or funnel data.
Steps:

1. Calculate funnel conversion rates at each stage.
2. Identify major drop-off points and analyze user behavior.
3. Segment users by traffic source, device, or demographics.
4. Propose A/B test ideas and estimate potential uplift.

5. Visualize funnel and key metrics in a dashboard or notebook.
   Learning outcomes: Product analytics, funnel metrics, business insights.
   Difficulty: Beginner–Intermediate.
   Extensions: Implement A/B test simulation or cohort analysis.

23. Predict Employee Attrition (Intermediate)

Objective: Predict which employees are likely to leave the company.
Tools: Python, pandas, scikit-learn.
Dataset: IBM HR Analytics Attrition dataset (Kaggle).
Steps:

1. Explore features like job satisfaction, salary, tenure.
2. Encode categorical features and handle imbalance.
3. Train classification models and interpret important features.
4. Evaluate with precision/recall focusing on minimizing false negatives.
5. Suggest HR interventions based on model insights.
   Learning outcomes: HR analytics, model interpretation, ethical considerations.
   Difficulty: Intermediate.
   Extensions: Build retention policy simulation.

---

24. Food Recognition from Images (Intermediate)

Objective: Classify types of food from images.
Tools: Python, Keras/PyTorch, transfer learning.
Dataset: Food-101 dataset.
Steps:

1. Preprocess images and use data augmentation.
2. Use transfer learning from a pre-trained CNN.
3. Train and validate, then evaluate accuracy/top-5 accuracy.
4. Create a simple app that identifies food from photos.
5. Discuss nutritional use-cases.
   Learning outcomes: Image classification and transfer learning.
   Difficulty: Intermediate.
   Extensions: Estimate calories from the dish using size/portion estimation.

25. Airline Delay Analysis & Prediction (Advanced)

Objective: Analyze flight delays and predict on-time performance.
Tools: Python, pandas, scikit-learn, XGBoost.

Dataset: U.S. Bureau of Transportation Statistics (BTS) or Kaggle flight datasets.
Steps:

1. Load historic flight data and merge with weather/airport data.
2. Explore delay patterns by carrier, route, and time.
3. Train classification/regression models to predict delays.
4. Evaluate models using ROC-AUC or RMSE and present actionable insights.
5. Visualize delay heatmaps and top risk factors.
   Learning outcomes: Data merging, feature engineering with external data, model evaluation.
   Difficulty: Advanced.
   Extensions: Real-time delay predictor with live weather feed.

26. Product Review Summarization (NLP) (Advanced)

Objective: Automatically summarize product reviews into short pros/cons.
Tools: Python, transformers (T5/BART), nltk.
Dataset: Amazon product reviews or Yelp.
Steps:

1. Collect reviews for specific products and clean text.
2. Fine-tune an abstractive summarization model or use extractive methods.
3. Evaluate with ROUGE scores and human validation.
4. Display summaries and typical pros/cons for products.
5. Discuss limitations of automated summarization.
   Learning outcomes: Text summarization, model fine-tuning, evaluation metrics.
   Difficulty: Advanced.
   Extensions: Aspect-based summarization (price, durability, battery).

27. Road Sign Recognition for Autonomous Driving (Advanced)

Objective: Detect and classify road signs in images.
Tools: Python, OpenCV, TensorFlow/Keras or PyTorch.
Dataset: German Traffic Sign Recognition Benchmark (GTSRB).
Steps:

1. Preprocess and augment sign images.
2. Train a CNN classifier and optionally a detection model (YOLO).
3. Evaluate with accuracy and mAP for detection models.
4. Optimize for speed and test on video frames.
5. Discuss deployment to embedded systems.
   Learning outcomes: Object detection/classification, model optimization.

Difficulty: Advanced.
Extensions: Real-time detection on dashcam video.

28. Automated Essay Scoring (Intermediate to Advanced)

Objective: Predict essay scores using textual features and models.
Tools: Python, nltk/spacy, scikit-learn, transformers.
Dataset: ASAP Automated Essay Scoring dataset (Kaggle).
Steps:

1. Extract features: readability, grammar errors, length, vocabulary richness.
2. Train regression or ordinal classification models.
3. Evaluate with quadratic weighted kappa or correlation.
4. Provide feedback on how scores are determined.
5. Address fairness and bias issues.
   Learning outcomes: NLP feature engineering, evaluation metrics for scoring.
   Difficulty: Intermediate–Advanced.
   Extensions: Provide automated feedback suggestions for improvement.

29. Image Captioning (Advanced)

Objective: Generate captions for images using encoder-decoder models.
Tools: Python, TensorFlow/Keras or PyTorch, CNN + LSTM/Transformer.
Dataset: Flickr8k/Flickr30k or MSCOCO (smaller for students).
Steps:

1. Extract image features using pre-trained CNN.
2. Train a sequence model (LSTM or Transformer) to generate captions.
3. Evaluate with BLEU/METEOR scores and human inspection.
4. Demonstrate captions for sample images and iterate.
5. Discuss challenges like diversity and hallucination.
   Learning outcomes: Multimodal learning, sequence modeling, evaluation of generative models.
   Difficulty: Advanced.
   Extensions: Add attention mechanism or beam search decoding.

30. Personal Finance Tracker & Insights (Beginner to Intermediate)

Objective: Build a tracker that analyzes spending patterns and suggests budget tips.
Tools: Python, pandas, matplotlib, Streamlit for UI.
Dataset: Personal or synthetic transaction data.
Steps:

1. Clean transactions and categorize expenses (or use regex/keywords).
2. Visualize monthly spending, category breakdown, and trends.
3. Create budget rules and calculate overspending alerts.
4. Provide simple savings suggestions based on trends.
5. Deploy with Streamlit for a personal dashboard.
   Learning outcomes: Data cleaning, time series aggregation, simple product analytics.
   Difficulty: Beginner–Intermediate.
   Extensions: Add forecasting for future expenses and goal tracking.

Tips for Completing a Strong Project

- **Start small, then iterate.** Don't try to build a perfect model first. Get a baseline working and improve.
- **Document your process.** Use clear markdown in notebooks: problem statement, data, EDA, modeling, results, and next steps.
- **Version control your code.** Use Git and push projects to GitHub — employers appreciate visible work.
- **Include visuals.** Plots and dashboards help communicate findings better than numbers alone.
- **Explain trade-offs.** Describe why you chose specific models or preprocessing steps.
- **Be honest about limitations.** Mention data quality issues, potential biases, and how the model might fail.
- **Prepare a short demo.** A simple README, a demo GIF, or a hosted Streamlit app makes your project accessible.

Suggested Tools & Learning Resources

- **Languages/Frameworks:** Python, R (optional).
- **Python libraries:** pandas, numpy, matplotlib, seaborn, scikit-learn, XGBoost/LightGBM, TensorFlow/Keras or PyTorch, nltk/spacy, transformers.
- **Deployment:** Streamlit, Flask, Heroku (or GitHub Pages for static).
- **Data sources:** Kaggle, UCI Machine Learning Repository, government open data portals, Google Dataset Search.
- **Version control & portfolios:** Git, GitHub, Jupyter Notebooks.

Must Read: 30 Data Analysis Project Ideas for Students

Conclusion

Working on projects is the best way for students to learn data science.

The 30 project ideas above cover a wide range of domains — education, healthcare, finance, computer vision, NLP, time series, and product analytics — and are designed to grow with your skills.

Start with beginner projects to build confidence, then tackle intermediate and advanced ideas that fit your interests.

For each project, focus on clear problem statements, reproducible code, strong visualizations, and concise explanations of results.

Share your work on GitHub, write short blog posts or README files, and practice explaining your projects in interviews. Consistent practice will transform these ideas into a portfolio that demonstrates not just technical ability but also real-world thinking and communication skills.

Good luck — pick a project, start today, and keep iterating!

Uncategorized

‹ 30 Digestive System Project Ideas for Kids 2026-27

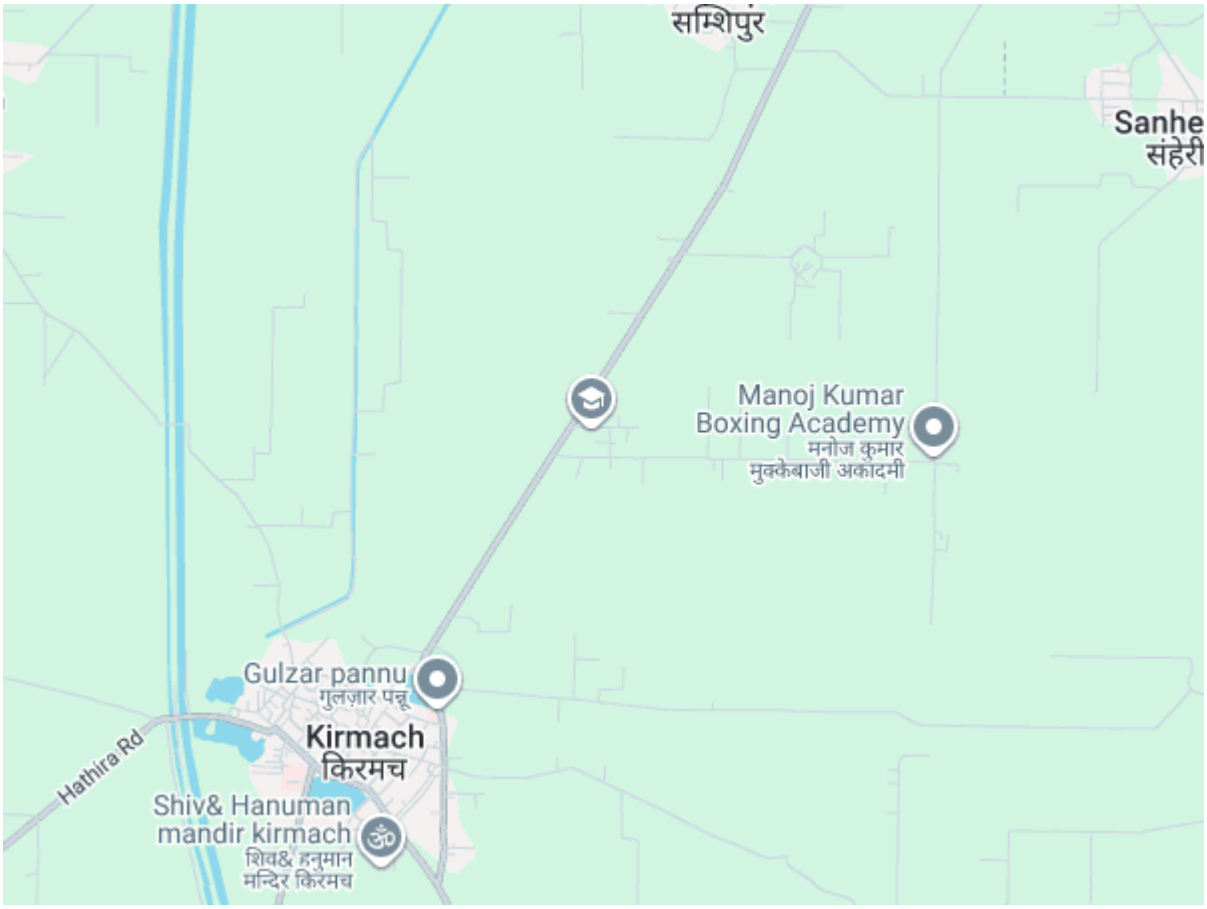
SKS TEAM

With years of experience, I work alongside a passionate group of educators and professionals to create a welcoming and supportive environment. At SKS International Gurukul, we focus on helping students grow both academically and personally, ensuring they have everything they need to succeed.

Leave a Comment

Logged in as admin1. Edit your profile. Log out? Required fields are marked *

Post Comment



Do not miss this experience!

ASK US ANY QUESTIONS

GET IN TOUCH

SKS International Gurukul - Kirmach Kurukshetra

About us

SKS International Gurukul, the best school in Kurukshetra, provides modern facilities, dedicated teachers, and engaging activities for Pre-nursery to 12th grade students.

Address

SKS International Gurukul, Near Nit, Kirmach Road Kurukshetra

Contact

School hours: 08:30am - 2:00pm

94160-73605 , 9315144282

sksinternationalgurukul@gmail.com